МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра технологий обработки и защиты информации

Курсовая работа

Ведение домашнего бюджета. Рекомендации по сокращению расходов

09.03.02 Информационные системы и технологии  
Защита информации в компьютерных системах

Обучающийся \_\_\_\_\_\_\_\_\_\_\_ Зеленев А.А., 3 курс, д/о

Обучающийся \_\_\_\_\_\_\_\_\_\_\_ Михно Д.А., 3 курс, д/о

Обучающийся \_\_\_\_\_\_\_\_\_\_\_ Ганигин К.И., 3 курс, д/о

Руководитель \_\_\_\_\_\_\_\_\_\_\_ Тарасов В.С., старший преподаватель

Воронеж 2021

Содержание

[Введение 4](#_Toc74861656)

[1 Постановка задачи. 5](#_Toc74861657)

[2 Анализ предметной области 6](#_Toc74861658)

[2.1 Глоссарий 6](#_Toc74861659)

[2.2 Анализ существующих решений 6](#_Toc74861660)

[2.2.1 Деньги ОК 6](#_Toc74861661)

[2.2.2 CoinKeeper 7](#_Toc74861662)

[2.2.3 Monefy 8](#_Toc74861663)

[2.2.4 Moneon 9](#_Toc74861664)

[2.2.5 Wallet 10](#_Toc74861665)

[2.3 Анализ поставленной задачи 11](#_Toc74861666)

[2.3.1 Use case 11](#_Toc74861667)

[2.3.2 Class diagram 12](#_Toc74861668)

[2.3.3 Sequence diagram 13](#_Toc74861669)

[2.3.4 State diagram 15](#_Toc74861670)

[2.3.5 Activity diagram 18](#_Toc74861671)

[2.3.6 Deploy diagram 19](#_Toc74861672)

[2.4 Анализ средств реализации 20](#_Toc74861673)

[3 Реализация 22](#_Toc74861674)

[3.1 Серверная часть приложения 22](#_Toc74861675)

[3.2 Графический интерфейс 22](#_Toc74861676)

[3.2.1 Главная страница 23](#_Toc74861677)

[4 Тестирование 26](#_Toc74861678)

[4.1 Smoke testing 26](#_Toc74861679)

[4.2 System testing 26](#_Toc74861680)

[5 Аналитика 27](#_Toc74861681)

[Заключение 28](#_Toc74861682)

Введение

Деньги – один из важнейших ресурсов в нашем мире. Люди трудятся, чтобы заработать, и тратят, чтобы получить блага: продукты, техника, образование, недвижимость. Отсюда возникает два вопроса: как заработать деньги и как их оптимально потратит. В рамках данного проекта мы собираемся ответить именно на второй вопрос. Ежедневно человек совершает около 10-15 операций по карте, зачастую даже не контролируя, сколько он тратит. В этой суете тяжело проконтролировать не просто месячные расходы, а даже дневные. Впрочем, проблему эту решили довольно просто. Ключ к спасению - ведение дневника расходов. Запись доходов, расходов, классификация трат, подведение итогов в конце месяц. В конце концов общий анализ и решения по дальнейшим действиям оставались на человеке. Но мы живем в век высоких технологий, мало кому захочется вести и хранить бумажную сводку покупок, да еще к тому же не актуальную. Вероятнее, люди предпочтут приложение, в которое можно будет быстро ввести информацию по тратам, чтобы оно обрабатывало информацию об изменениях бюджета и показывало графики, прогнозы и рекомендации об улучшении текущего финансового положения. Такому приложению и посвящена данная курсовая работа.

# Постановка задачи.

Цель данной курсовой работы – создать веб-приложение для ведения и контроля семейного бюджета. Бюджет должен представлять из себя статьи доходов и статьи расходов, соответственно, в процессе ведения домашнего бюджета необходимо вести учет данных показателей, чтобы потом соотносить фактические траты с планируемыми.

Веб-приложение должно обладать следующей функциональностью:

* Регистрация расходов/доходов.
* Составление плана расходов на месяц.
* Предоставление рекомендаций по снижению расходов.
* Создание финансовых целей.
* Расчет размера финансового резерва.
* Формирование отчета по расходам за определенный период времени.

Для этого потребуется решить следующие задачи:

* Изучить предметную область
* Провести анализ существующих решений
* Определить какие будут использованы средства и методы для данного приложения
* Разработать приложение, которое реализует вышеперечисленные функциональности.

# Анализ предметной области

## Глоссарий

Расходы - затраты, которые ведут к уменьшению экономических выгод в результате выбытия денежных средств.

Доходы - денежные средства, полученные физическим лицом в результате какой-либо деятельности за определённый период времени.

Счет – учетная позиция, создаваемая для хранения информация о текущем денежном состоянии пользователя.

Бюджет - финансовый план определённого субъекта (семьи, бизнеса, организации, государства и т. д.), устанавливаемый на определённый период времени.

Семейный бюджет - план доходов, расходов и накоплений, описывающий возможности всех членов семьи в определённый период времени.

Виджет - примитив графического интерфейса пользователя, имеющий стандартный внешний вид и выполняющий стандартные действия.

Веб-приложение - клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика вебприложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети.

## Анализ существующих решений

### Деньги ОК

Количество установок: более 100 000. Имеется разовая подписка (PRO-версия), расширяющая функциональность.

Тенденция развития: Стабильно 1-2 раза в месяц разработчики выпускают обновления с исправленными ошибками. Так же приложение периодически получает масштабные обновления, но связаны они чаще всего с внешним видом.

Оценка с позиции клиента:

Приложение имеет оценку 4.8 в AppStore. Огромное количество пользователей IOS хвалит интерфейс за его простоту и удобство, за возможность синхронизации и перевод валюты. Однако сообщество Android устройств встретило приложение менее тепло. Оценка 4.2, вечные жалобы на неисправности, никаких ответов от разработчиков.

Функциональности:

* Учет расходов и доходов.
* Неограниченное число счетов и статей (категорий).
* Подстатьи.
* Переводы между счетами.
* Дневные, месячные и годовые отчеты.
* Синхронизация.
* Дневные, недельные, годовые отчеты (PRO-версия).
* Составление бюджета (PRO-версия).
* Виджет для ввода расходов и доходов (PRO-версия).

### CoinKeeper

Количество установок: более 1 000 000. Имеется годовая и месячная подписки, а также разовая (навсегда).

Тенденция развития: Стабильно 1-2 раза в месяц разработчики выпускают обновления с исправленными ошибками и добавлением новой функциональности. Каких-то масштабных обновлений, приложение не получало с даты выхода.

Оценка с позиции клиента:

AppStore: Оценка 4.6

GooglePlay: Оценка 4.5

Несмотря на высокую оценку, пользователи постоянно жалуются на разные ошибки, рекламу после покупки подписки и др.

Функциональности:

* Автоматически распознает СМС-сообщения от любых банков.
* Поддержка всех мировых валют.
* Защита данных паролем.
* Общий профиль для семьи.

### Monefy

Количество установок: более 5 000 000. Имеется разовая подписка (PRO-версия) расширяющая функциональность.

Тенденция развития: Разработчики выпускают каждые 4-5 дней обновления с исправлением ошибок, улучшением интерфейса.

Оценка с позиции клиента:

AppStore: Оценка 4.8

GooglePlay: Оценка 4.5

Пользователи хвалят приложение практически со всех сторон. Когда клиенты сообщают об ошибках разработчики быстро реагируют на отзыв.

Функциональности:

* Можно просматривать общий журнал расходов или детализировать данные.
* Безопасная синхронизация через учетную запись Google Drive или Dropbox.
* Можно контролировать кредиты, ежедневные и ежемесячные траты.
* Поддержка различных валют.
* Виджеты для быстрого доступа.
* Можно управлять категориями. При этом изменения синхронизируются между устройствами.
* Простое резервное копирование и экспорт данных.
* Можно использовать несколько учетных записей и считать расходы вместе с членами семьи.
* Простой подсчет расходов и доходов при помощи встроенного калькулятора.
* Возможность вести учет счетов в разных валютах (PRO-версия).
* Наличие пароля обеспечит безопасность данных (PRO-версия).

### Moneon

Количество установок: более 50 000. Имеется месячная и годовая расширяющая функциональность подписка.

Тенденция развития: Разработчики быстро решают проблемы. 2-3 обновления в месяц с исправлением ошибок и каждые 6-7 месяцев глобальные обновления, которые расширяют функционал программы.

Оценка с позиции клиента:

AppStore: Оценка 4.7

GooglePlay: Оценка 3.9

Клиенты IOS хвалят приложения со всех сторон и дают советы разработчикам. Клиенты Android очень сильно критикуют приложение, разработчики создают иллюзию реакции на отзывы, но не сильно заинтересованы в своих пользователях.

Функциональности:

* Учет личных расходов и расчет бюджета.
* Трекер расходов и контроль доходов.
* Совместный онлайн кошелек с семьей или друзьями (PRO версия).
* Личный бюджет, его планирование.
* Категории для анализа расходов и доходов.
* Множество валют на выбор.
* Выбор периода для финансового анализа.
* Защита паролем.

### Wallet

Количество установок: более 5 000 000. Имеется месячная, годовая и разовая покупка полной функциональности приложения.

Тенденция развития: Разработчики выпускают 3-4 обновления с исправленными ошибками в месяц, каждые 4-6 месяцев выходят крупные обновления, расширяющие функционал.

Оценка с позиции клиента:

AppStore: Оценка 4.0

GooglePlay: Оценка 4.7

Пользователи Android высоко оценивают приложение, при этом не скупы на критику и очень категорично высказываются о недоработках. Разработчики выборочно реагируют на отзывы.

Пользователи IOS приняли приложение чуть хуже, однако разработчики быстро адаптировали приложение под сообщество.

Функциональность:

* Импорт банковских транзакций.
* Экспорт в CSV/XLS/PDF.
* Умный помощник.
* Поддержка нескольких валют.
* Карты лояльности.
* Автоматическая синхронизация с банком (PRO-версия).
* Чеки и гарантии.
* Категории и шаблоны.
* Списки покупок.
* Управление долгами.

## Анализ поставленной задачи

### Use case

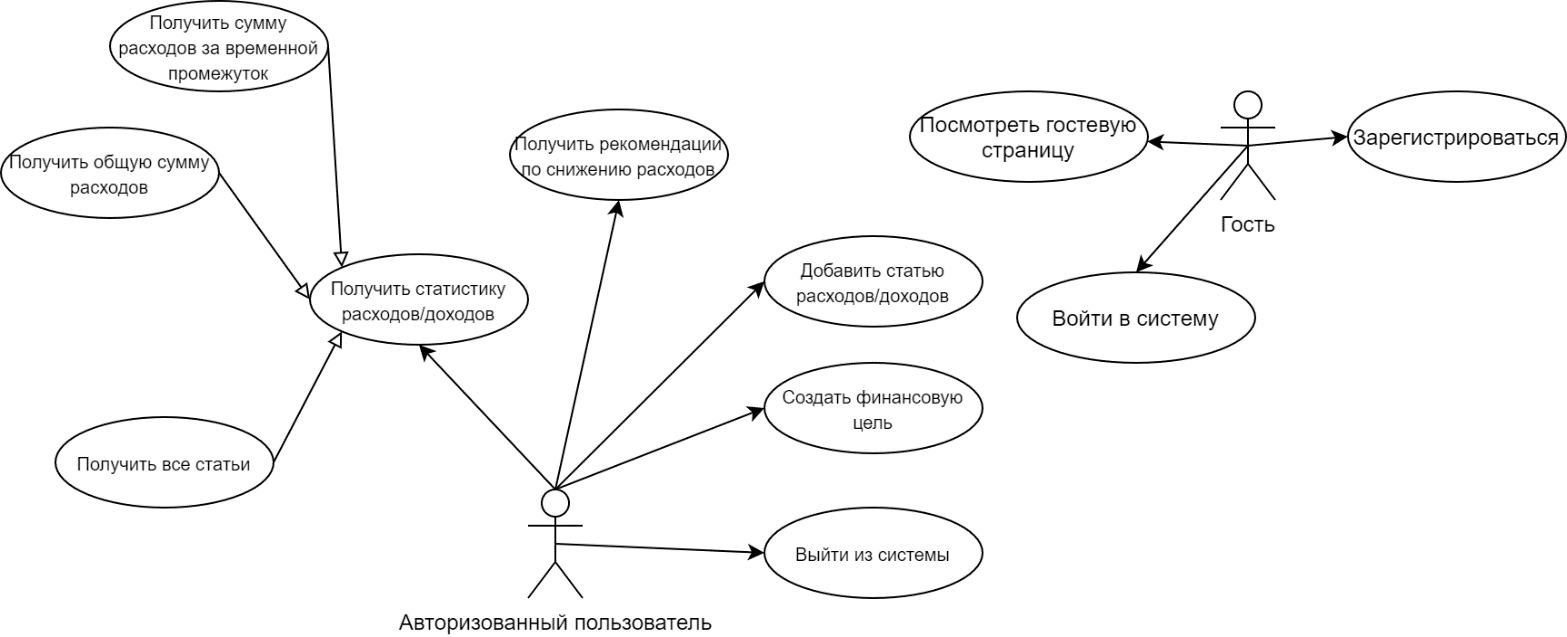


Рисунок 1 – Use case

При работе с приложением Budger неавторизованный пользователь, именуемый далее гостем, имеет следующие возможности:

* Просмотр гостевой страницы
* Зарегистрироваться
* Войти в систему

В свою очередь, авторизованный пользователь данного приложения, то есть пользователь, который зарегистрировался и вошел в систему под своими учетными данными, обладает приведенными ниже возможностями:

* Добавить расходы.
* Добавить финансовые цели.
* Получить рекомендации по расходам.
* Получить статистику расходов.
* Выйти из системы.

### Class diagram

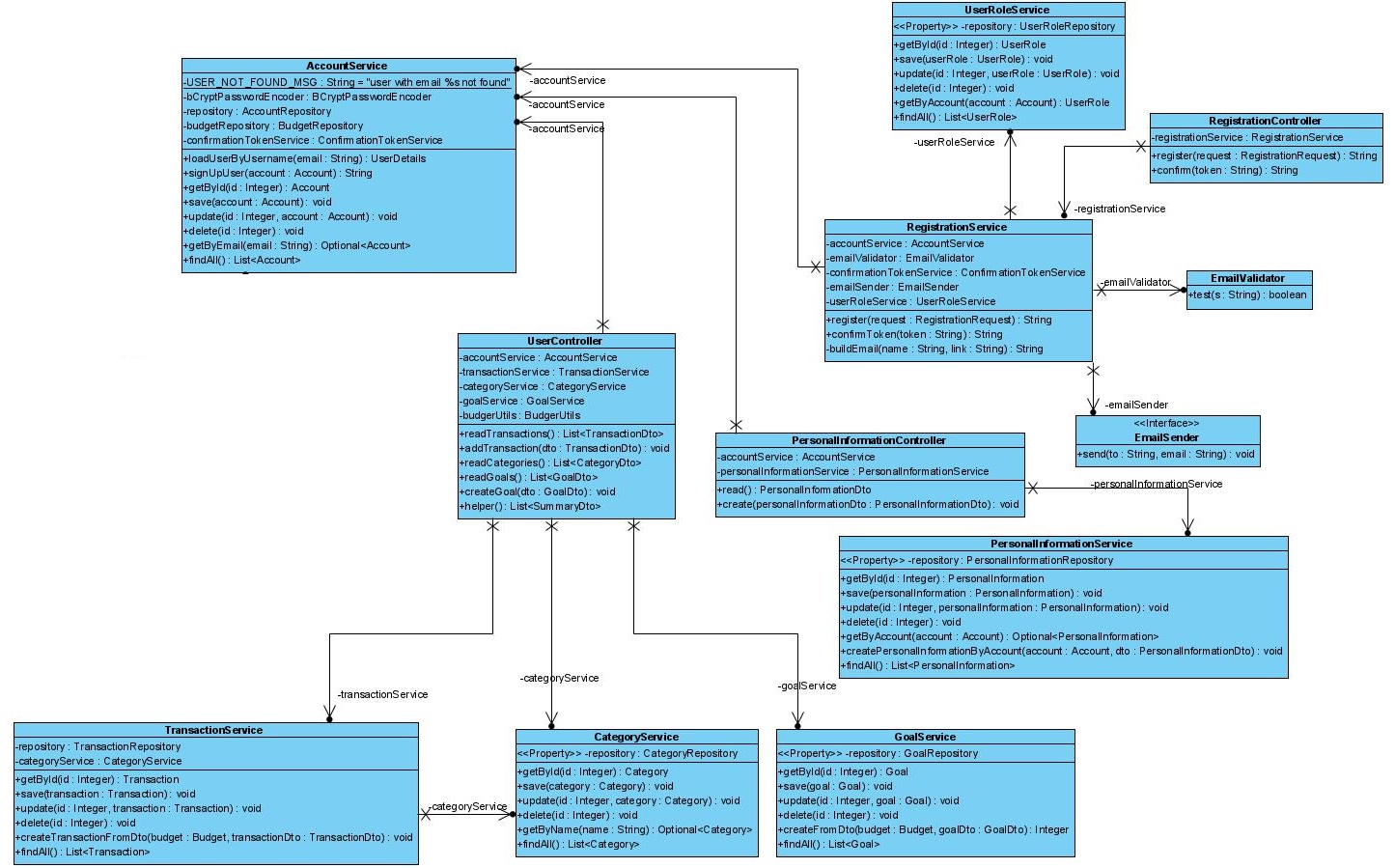


Рисунок 2 – Class diagram

Серверная часть приложения организована следующим образом: классы AccountService, CategoryService, RegistrationService и т.д. обеспечивают получение и запись данных в БД. Для общения с сервером существуют UserController, PersonalInformationController и RedistrationController. Они реализуют основные методы получение данных, которые запрашивает клиент. Контроллеры используют соответствующие сервисы, которые реализуют основной набор методов для работы с репозиториями, которые работают с базой данных.

### Sequence diagram

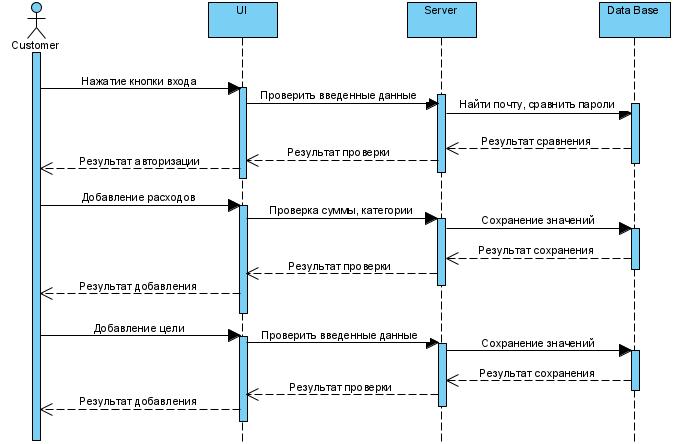


Рисунок 3 – Sequence diagram 1

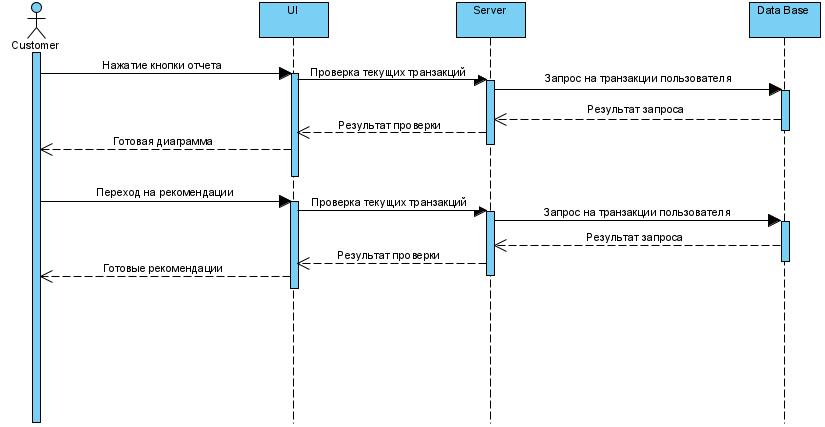


Рисунок 4 – Sequence diagram 2

На приведенном рисунке продемонстрирована диаграмма последовательностей и взаимодействий, на которых показаны отношения между объектами при входе в систему и при создании семейного профиля.

Пользователь использует графический интерфейс, чтобы войти в систему. Интерфейс в свою очередь передает запрос на добавление серверу, а сервер впоследствии проверяет и сравнивает данные введенные пользователем и данные в базе данных. База данных возвращает результат запроса, и в случае, если запрос прошел неудачно, сервер передает сообщение об ошибке графическому интерфейсу, а тот в свою очередь отображает сообщение об ошибке.

### State diagram

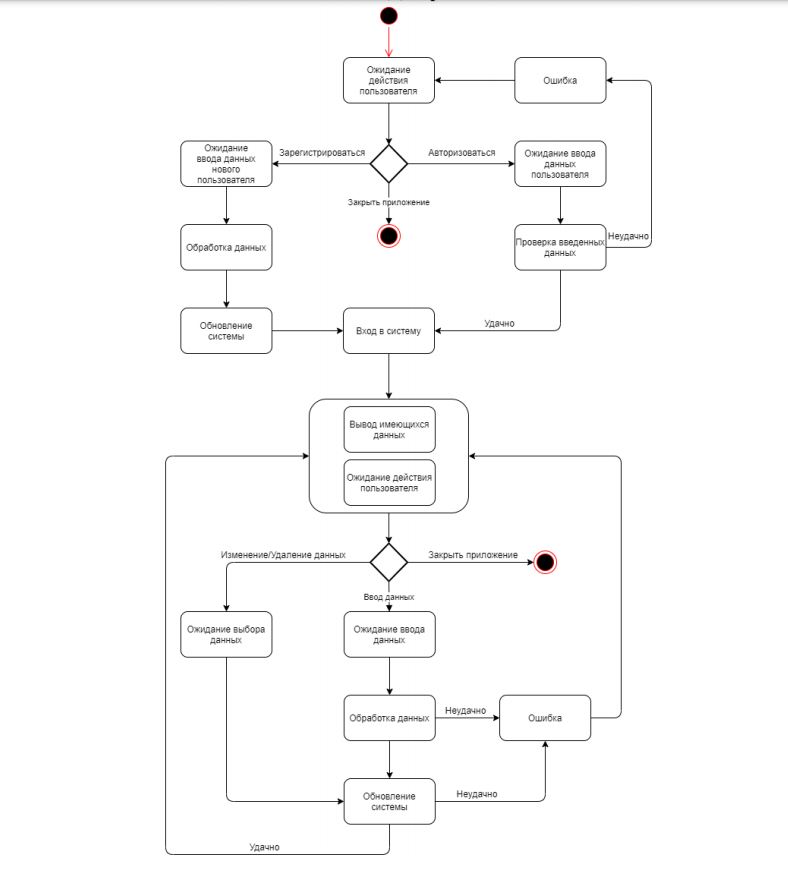


Рисунок 5 – State diagram

Данная диаграмма состояний отражает все возможные состояния системы, в которых она может находиться. При входе в приложение, система находится в состоянии ожидания действий пользователя. Из этого состояния она может перейти в одно из следующих в зависимости от конкретных действий пользователя:

* Зарегистрироваться.
* Авторизоваться.
* Закрыть приложение.

В случае, если пользователь принял решение зарегистрироваться, система переходит в состояние ожидания ввода данных нового пользователя, затем введенные пользователем данные обрабатываются, то есть система находится в состоянии обработки данных, после чего она (система) обновляется и в последствии происходит вход в систему.

При авторизации пользователя система также входит в состояние ожидания ввода данных пользователя, а затем в состояние проверки введенных данных и, если введенные данные корректны, происходит вход в систему. В противном случае система входит в состояние ошибки и возвращается в начальное состояние ожиданий действий пользователя как при входе в систему. Если пользователь закроет приложение, произойдет выход из системы, то есть из самого приложения.

После входа в систему, приложение переходит в состояние ожидания действия пользователя, при котором у него (пользователя) есть следующие возможности:

* Изменение/Удаление данных
* Ввод данных
* Закрыть приложение

При изменении или удалении данных, система переходит в состояние ожидания выбора тех существующих в базе данных, которые хочет изменить или удалить пользователь приложения. После чего система входит в состояние обновления данных. Если данные обновились успешно, система переходит в состояние ожидания тех же действий пользователя, как и после входа в систему, иначе — произойдет ошибка. Если пользователь выберет ввод новых данных, система будет ожидать ввода данных, которые впоследствии нужно будет обработать. В случае удачной обработки введенной пользователем информации, произойдет обновление системы, однако, если полученные данные были некорректны, приложение выдаст ошибку и предложит пользователю заново их ввести. При успешном обновлении системы, приложение будет ожидать дальнейших действий пользователя, в противном случае произойдет ошибка, информация не сохранится и пользователю приложения будет необходимо проделать действия заново.

### Activity diagram

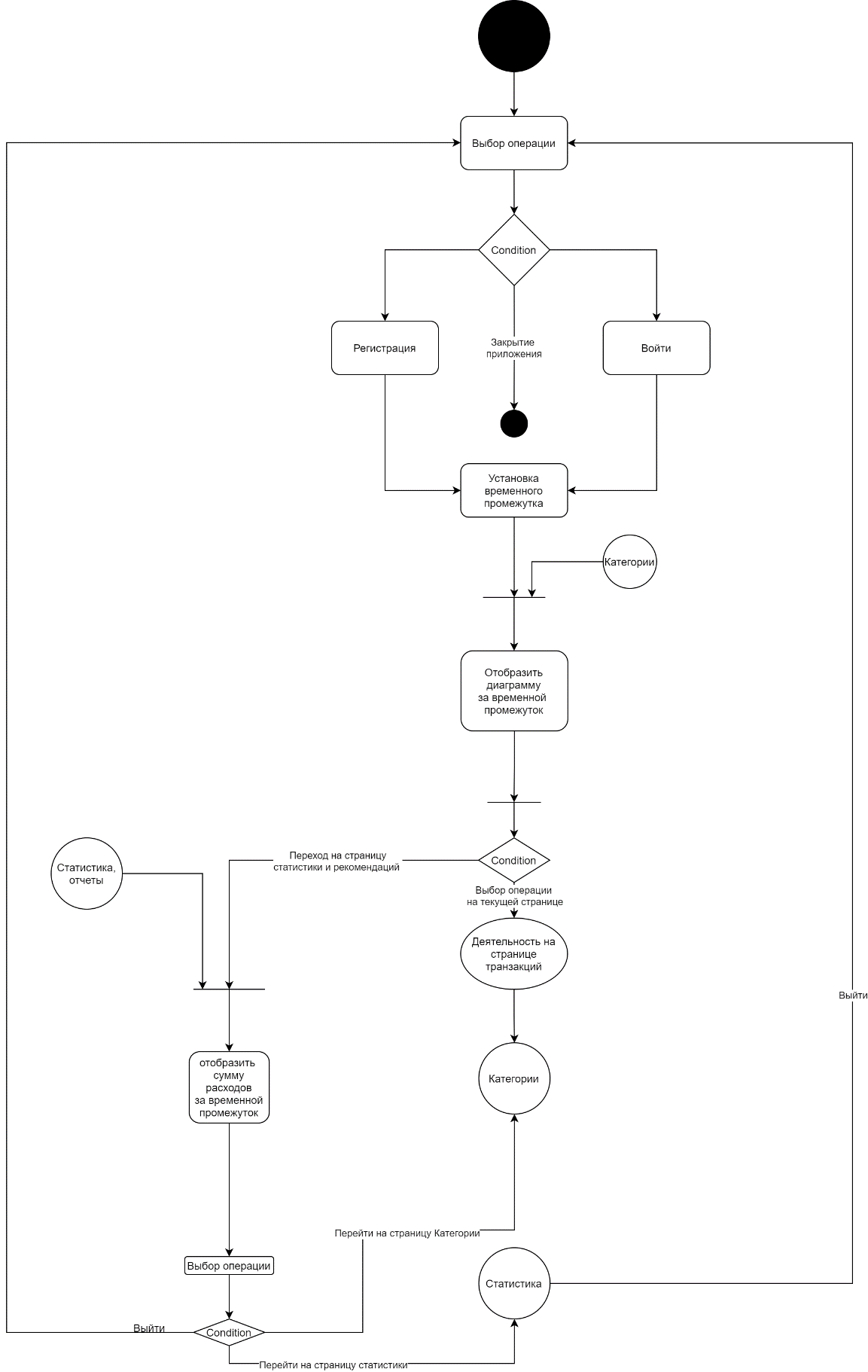


Рисунок 6 – Activity diagram

### Deploy diagram

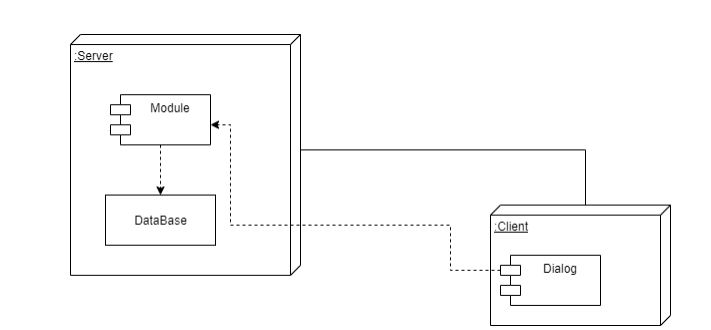


Рисунок 7 – Deploy diagram

Представленная нами диаграмма развертывания служит для демонстрации существующих аппаратных и программных компонентов приложения и взаимодействия между ними. На стороне сервера реализован весь необходимый функционал для успешной работы разрабатываемой системы, который неразрывно связан с базой данных, расположенной здесь же на сервере. На клиентской стороне приложения находится диалоговое окно, которое необходимо для обработки всех действий пользователя и передачи введенных или выбранных данных.

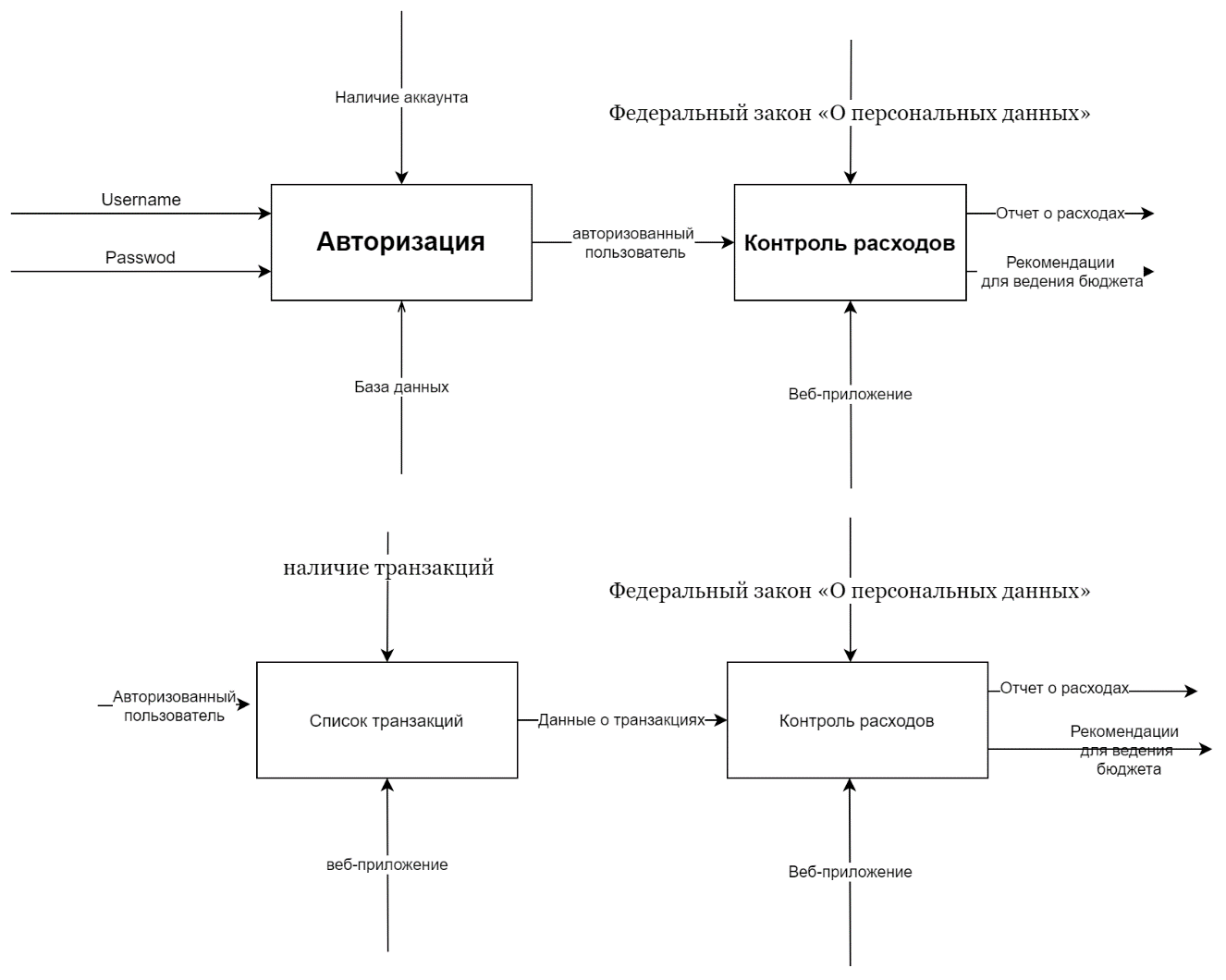


Рисунок 8 – IDEF0

## Анализ средств реализации

Для реализации и разработки приложения Budger были выбраны следующие технологии:

Java в качестве основного языка программирования вследствие следующих причин:

Все члены команды имеют опыт работы с языком программирования

Подходит для разработки веб приложений.

JavaScript как встраиваемый язык для программного доступа к объектам приложений. Нашел применение в приложении как язык для придания интерактивности веб-страницам

HTML как стандартизированный язык текстовой разметки

CSS в качестве формального языка описания внешнего вида веб-приложения

В качестве СУБД был выбран PostgreSQL по нескольким причинам:

* Масштабируемость.
* Многофункционально
* Простота в использовании

Взаимодействие между Front-end и Back-end осуществляется за счет REST API

Обмен данными между Front-end и Back-end происходит с помощью передачи JSON файлов.

# Реализация

## Серверная часть приложения

Серверная часть приложения реализована с использованием языка Java, а также базы данных PostgreSQL. Серверная часть приложения предоставляет клиентской части доступ к набору методов, необходимые для реализации основных клиентских сценариев. Данный набор методов называется API. Организация взаимодействия между клиентом и сервером с использованием данного подхода позволяет иметь возможность доступа к набору методов данного приложения из других систем.

## Графический интерфейс

Клиентская часть представляет собой SPA-приложение, реализованное с помощью javascript библиотеки React. Для разметки использовалось расширение javascript – jsx. Стили реализованы с помощью языка css.

### Главная страница

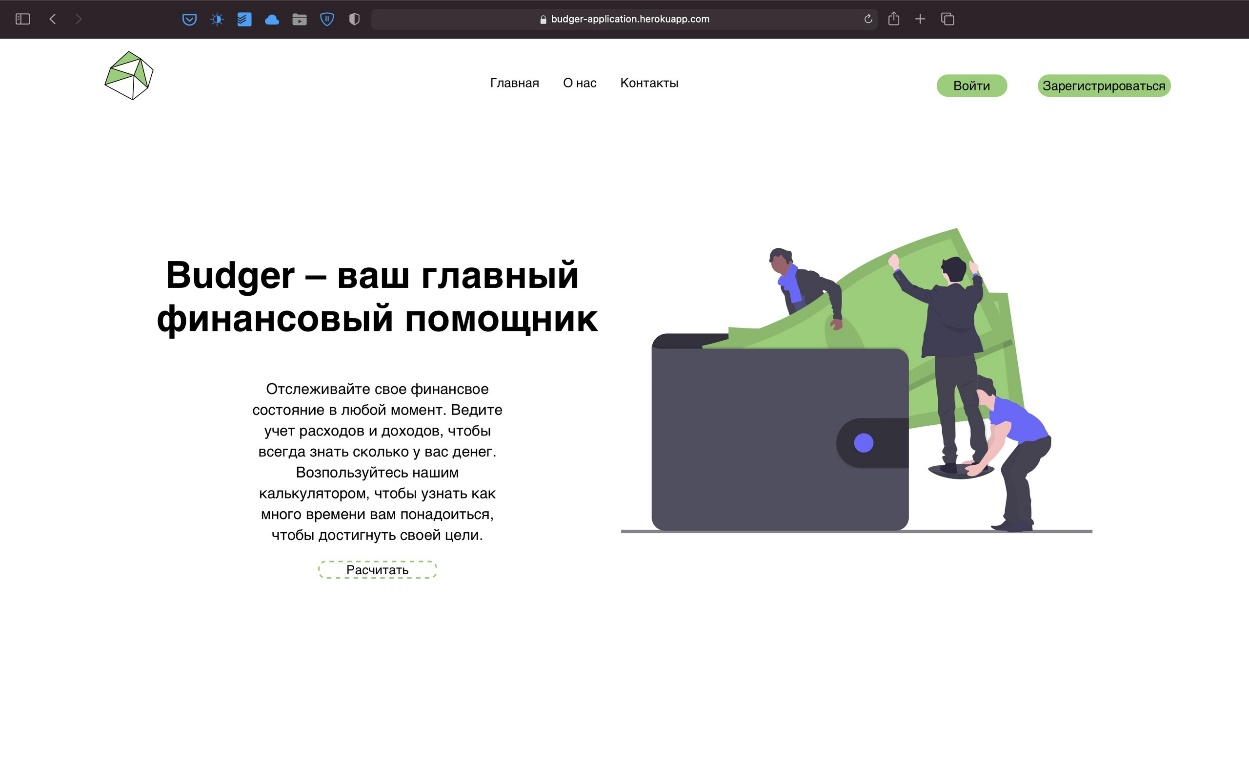


Рисунок 9 – Главная страница

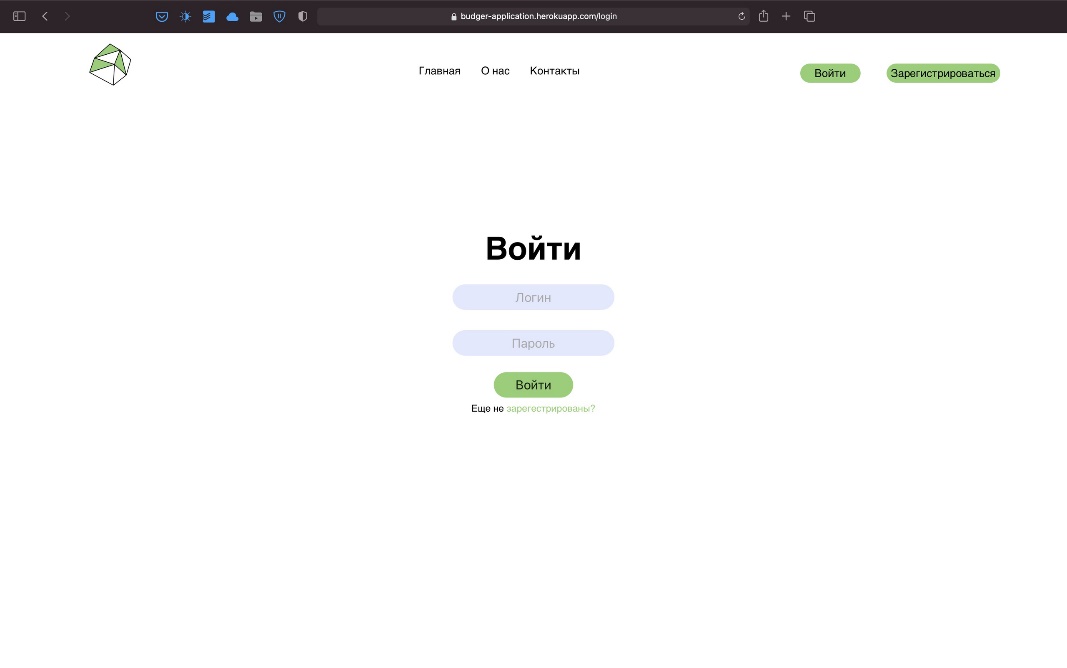


Рисунок 10 – Страница логина

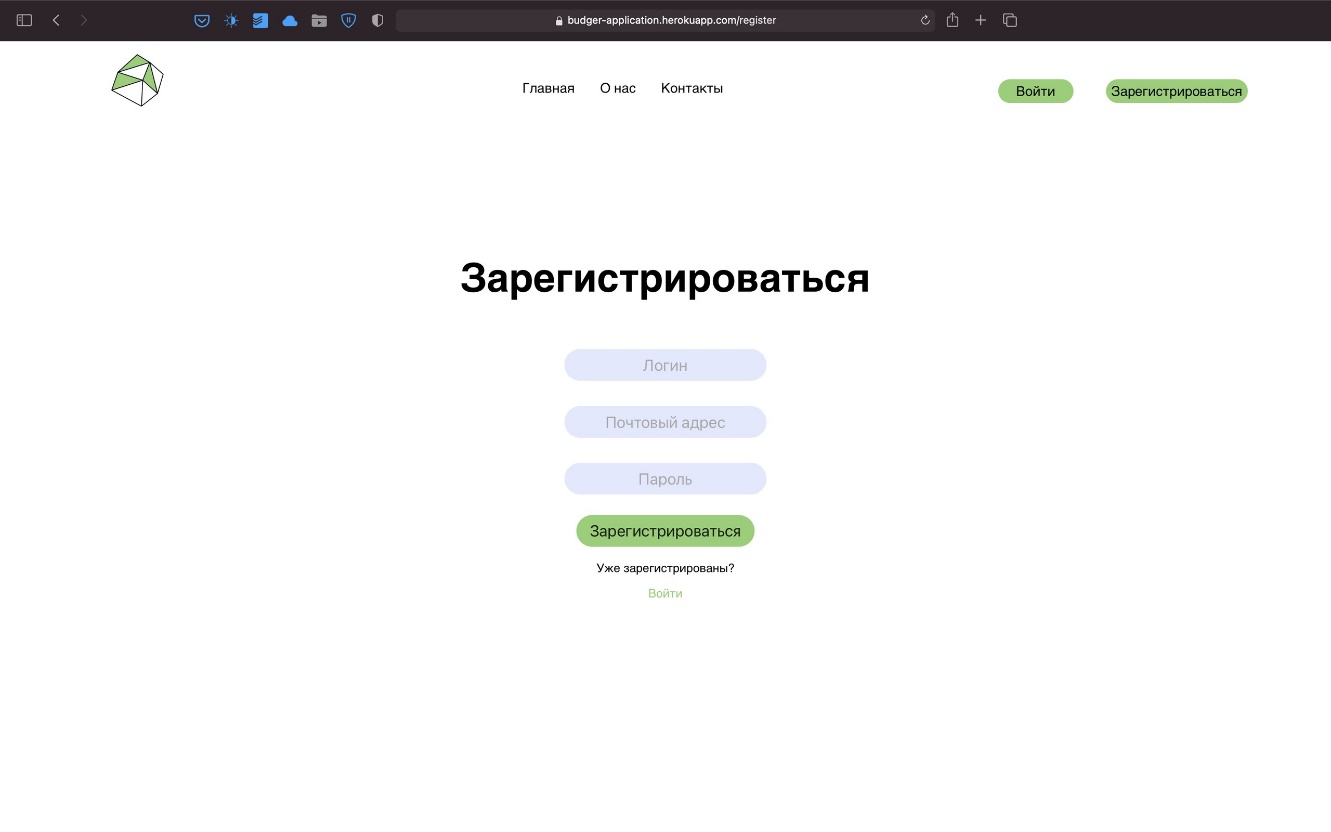


Рисунок 11 – Страница регистрации

Для регистрации необходимо ввести имя пользователя, почту и пароль. Если пользователя с таким именем и почтой в системе нет, то будет создан новый.

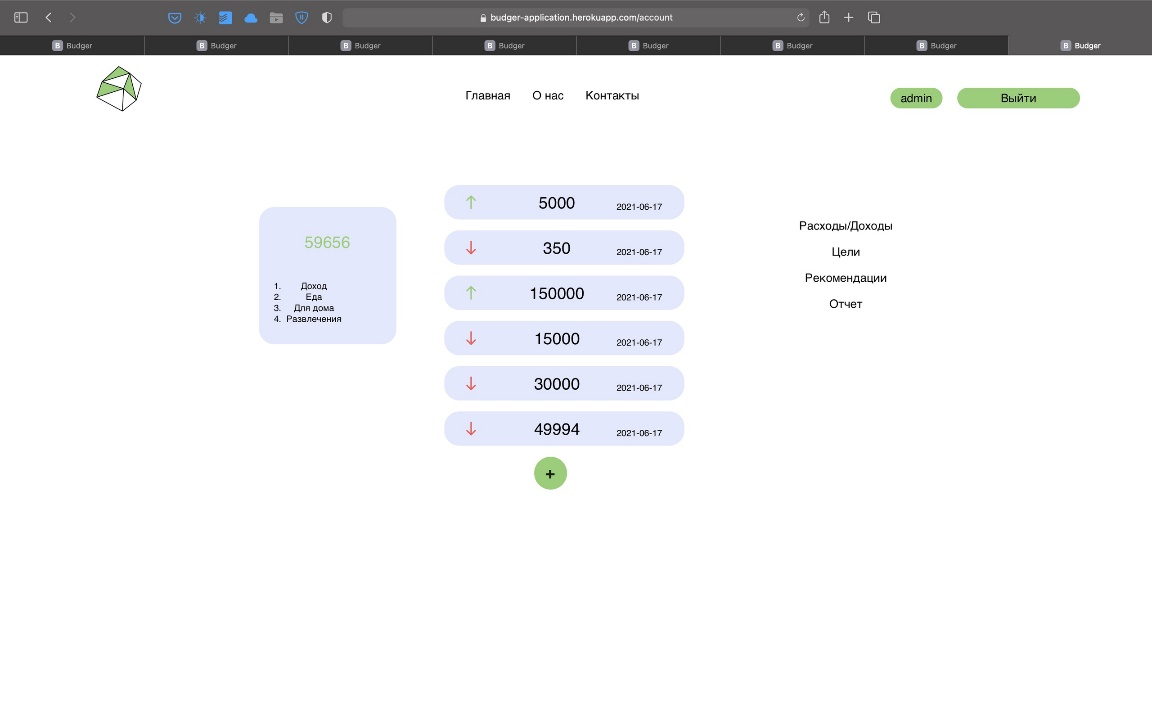


Рисунок 12 – Страница транзакций

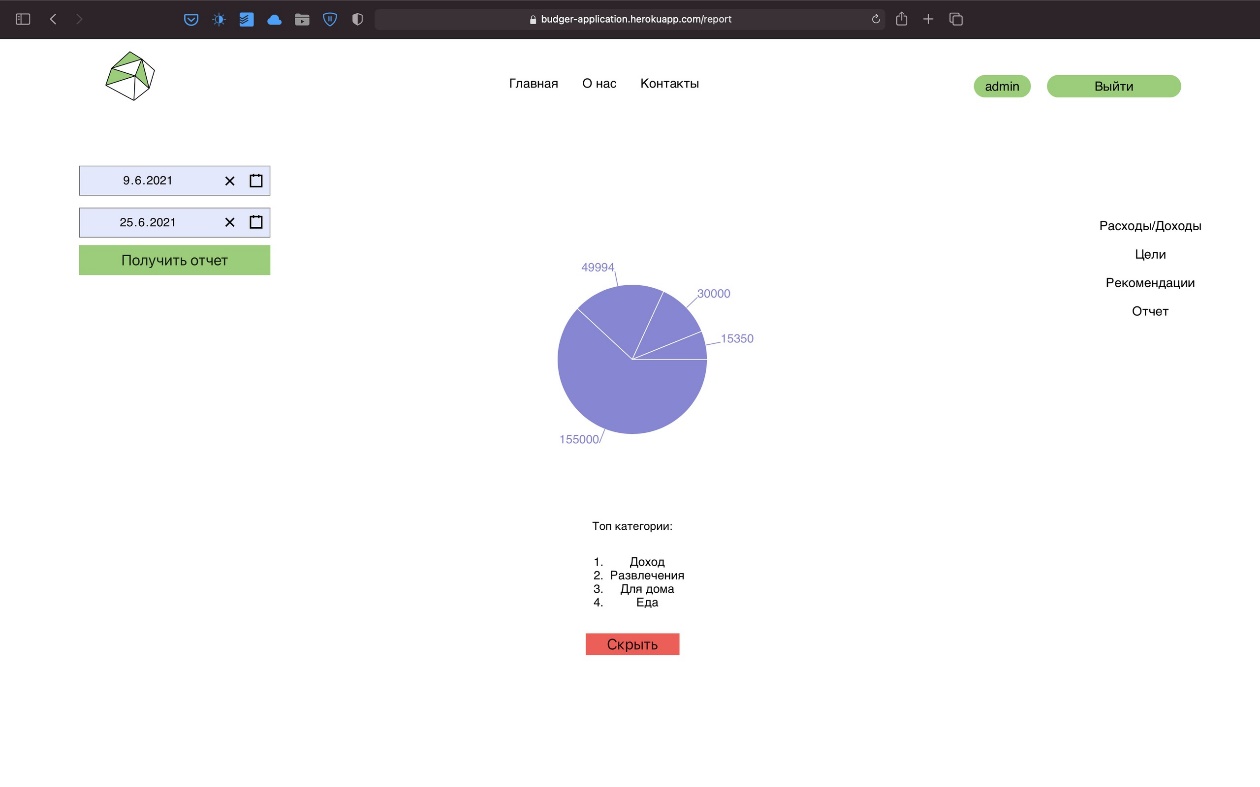


Рисунок 13 – Страница отчета

# Тестирование

Во процессе разработки приложения были созданы тестовые процедуры. По завершении разработки приложения, было проведено тестирование по работе всего функционала веб-приложения Budger для проверки соответствия ожидаемого результата работы, указанного в техническом задании, с фактическим. Были выбран следующие процедуры ручного тестирования:

* System test
* Smoke test

## Smoke testing

Smoke testing представляет собой короткий цикл тестов, выполняемых для подтверждения того, что приложение выполняет основные функции. Данная процедура была использована при каждой итерации деплоя приложения на хостинг.

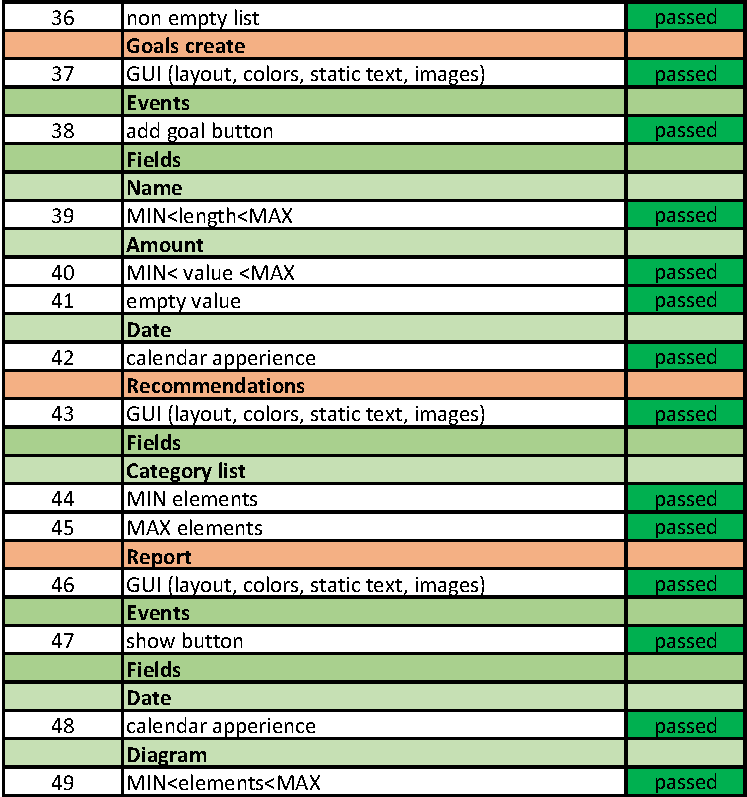
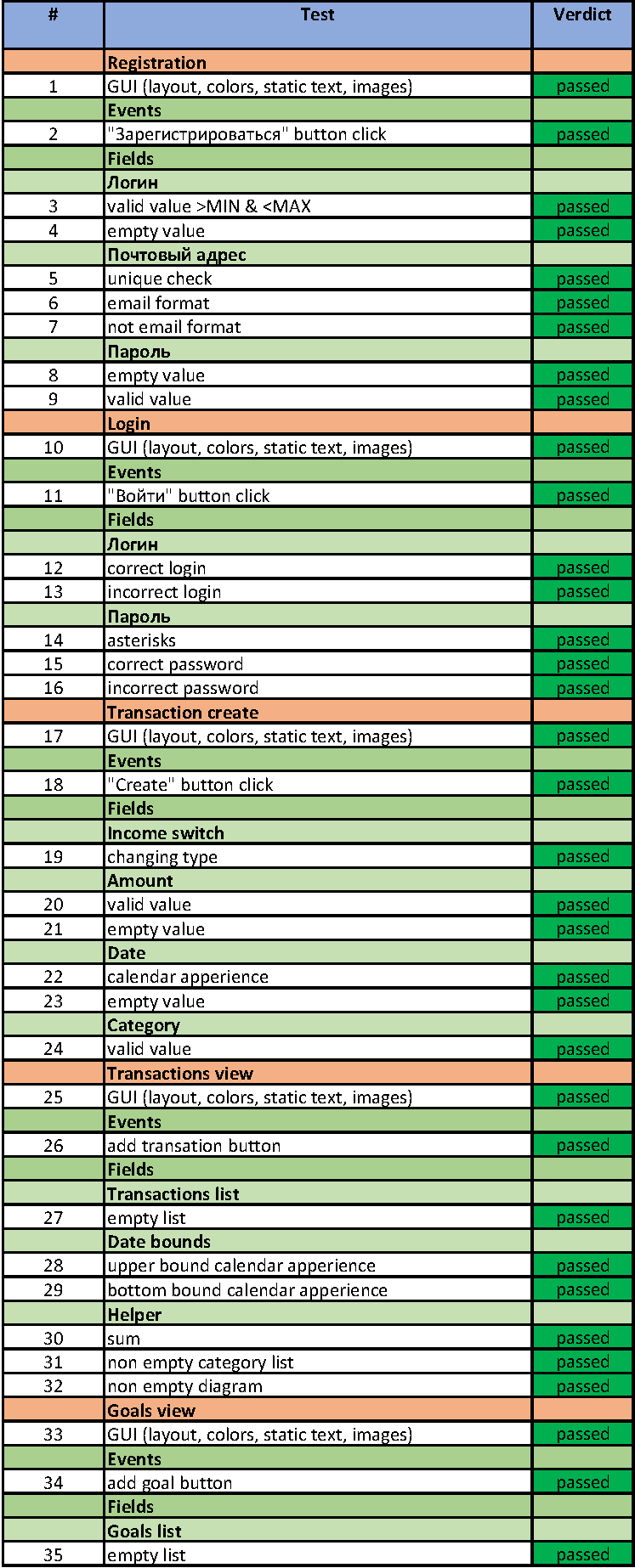


Рисунок 14 – Smoke testing

## System testing

System testing использовалась при первом деплое, чтобы подтвердить работоспособность всех функций приложения.

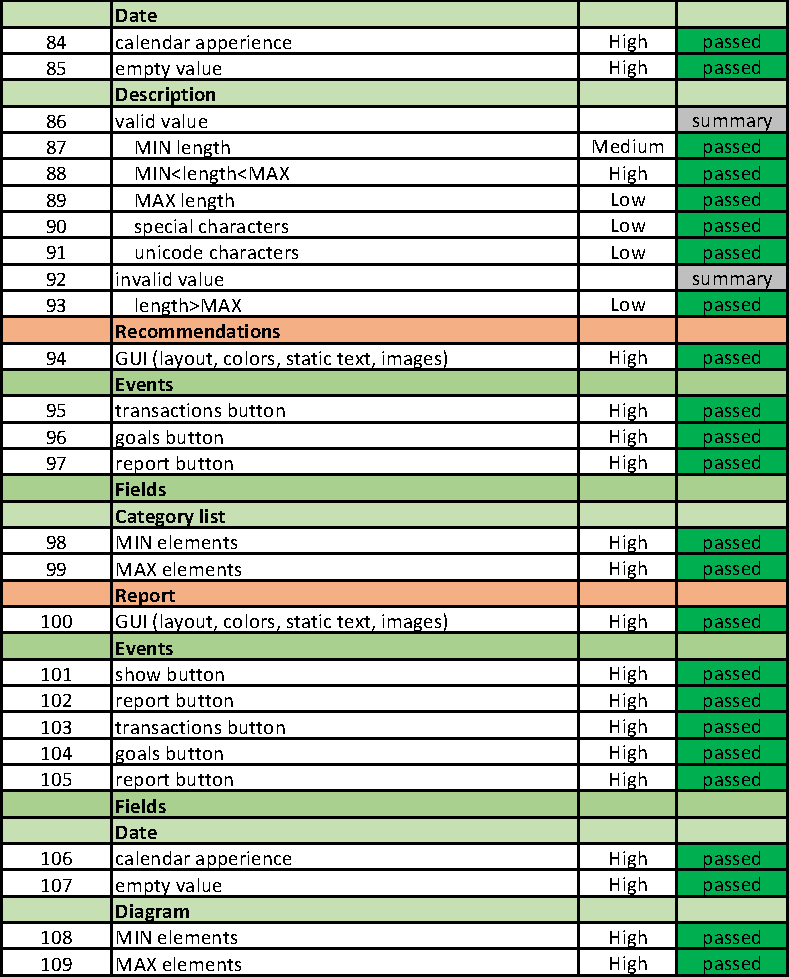
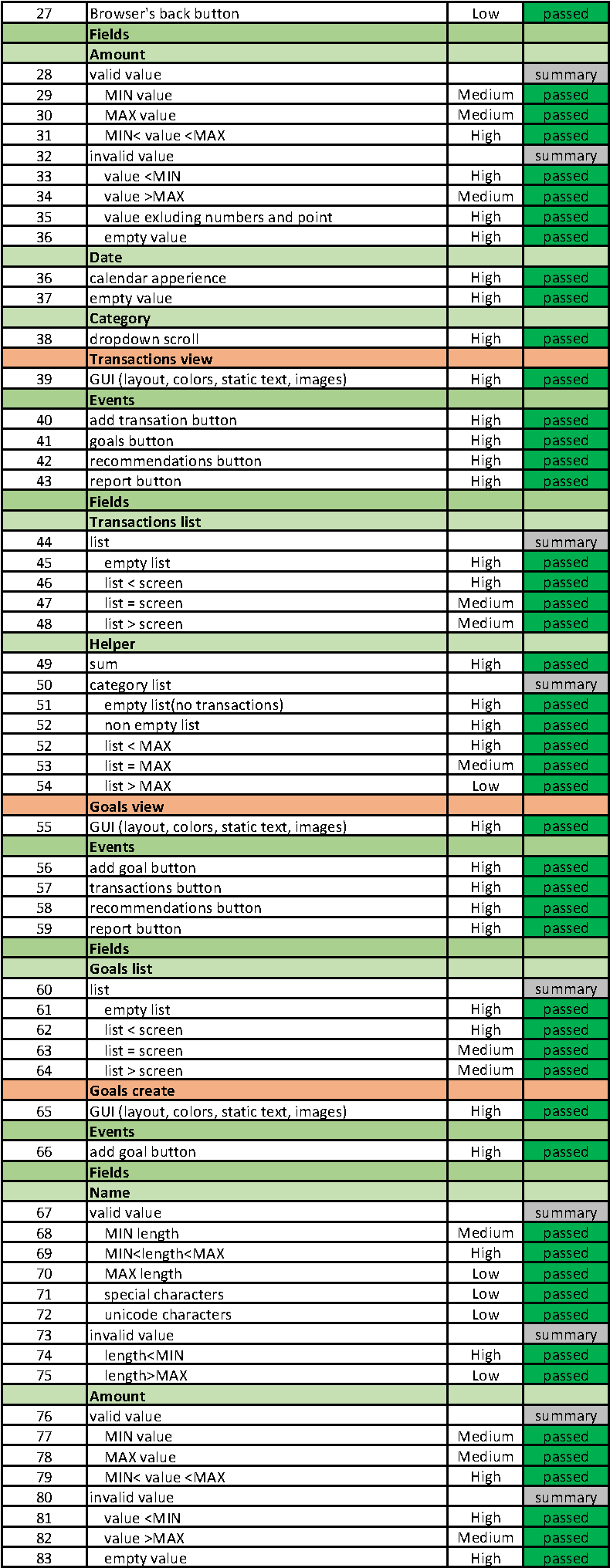
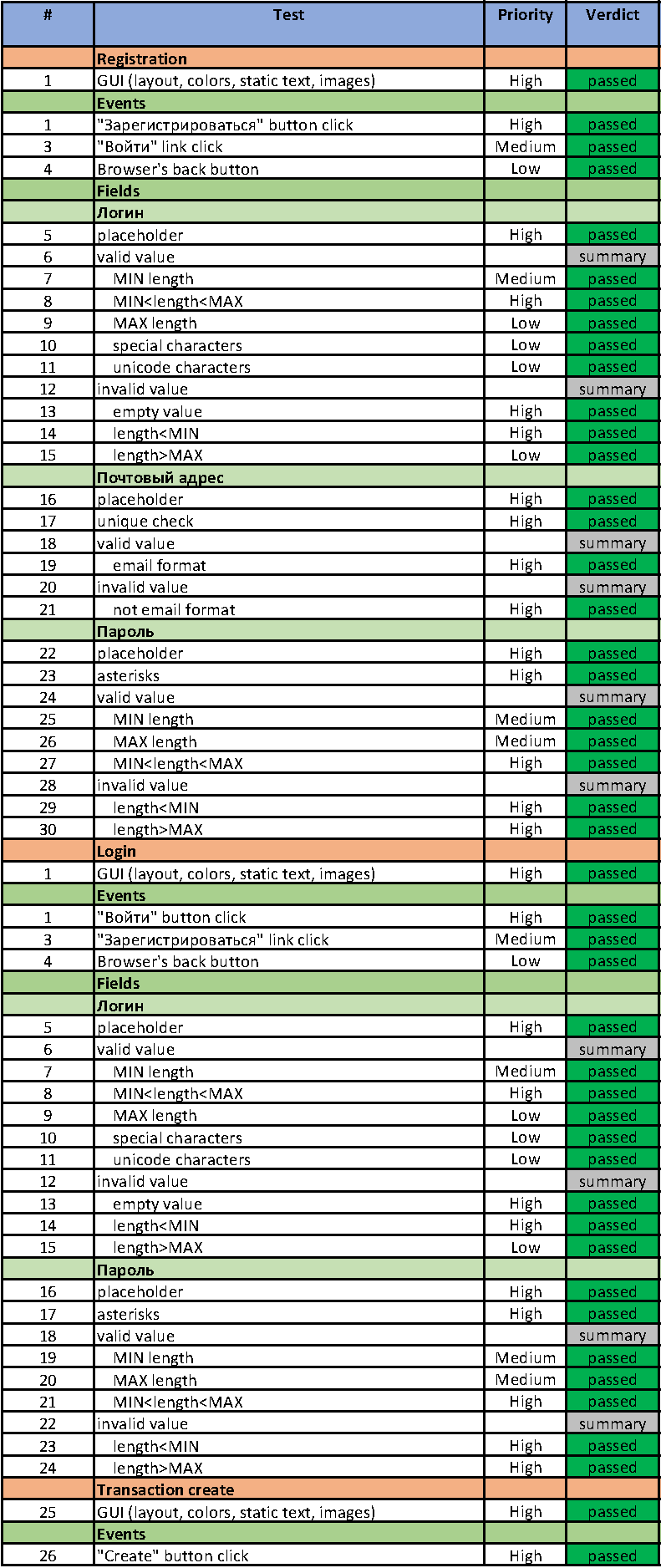


Рисунок 15 – System testing

# Аналитика

Для аналитики приложения использовался сервис Яндекс.Метрика. С его помощью можно отслеживать активность пользователь за разные промежутки времени, посещаемость определенных роутов и др.

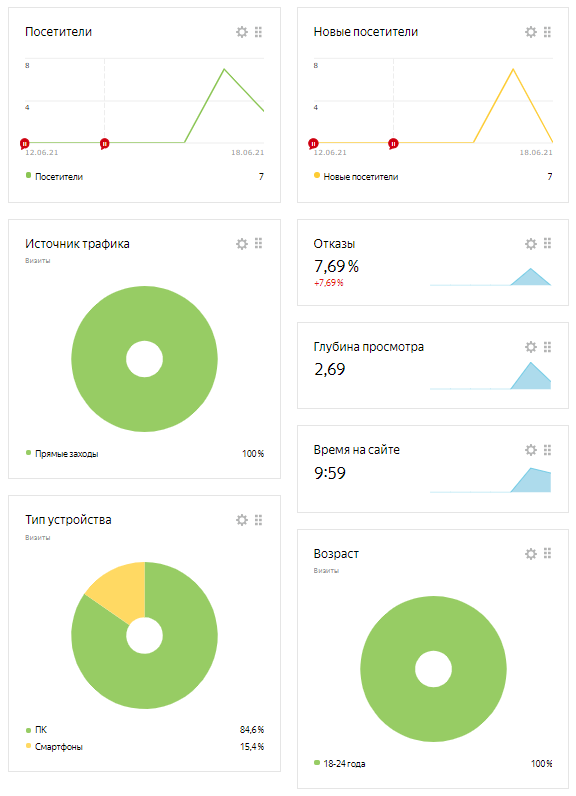


Рисунок 16 – Сводка Яндекс.Метрики

Так же средствами данного сервиса были созданы три основных пользовательских сценария, чтобы определить насколько пользователи заинтересованы в тех или иных функциях.

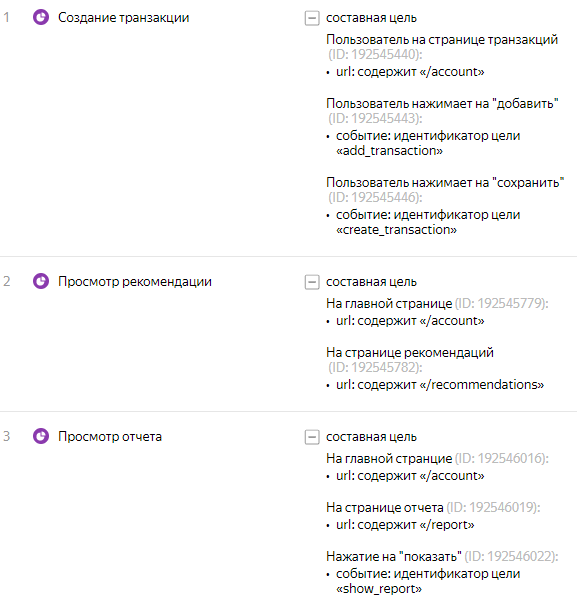


Рисунок 17 – 3 воронкки коверсии

Заключение

В результате проделанной работы было реализовано приложение Budger, которое позволяет пользователю осуществлять регистрацию и систематизацию своих расходов, а также просматривать различную статистику за определенный временной промежуток. Веб-приложение Budger обладает визуальным интерактивным интерфейсом, который необходим для взаимодействия пользователя с приложением и позволяет выводить на экран всю нужную информацию для удобства работы пользователя с сервисом. Были выполнены следующие задачи:

* Разработана база данных для хранения информации о пользователе, его расходах, доходах, расположенная на удаленном сервере.
* Разработан Back-end приложения необходимы для работы всего функционала сервиса развернутый на удаленном сервере приложения
* Разработан Front-end приложения, который необходим для взаимодействия пользователя с сервисом приложения
* С помощью API была реализована связь между Back-end и Front-end приложения